The kvsetkeys package

Heiko Oberdiek∗

<heiko.oberdiek at googlemail.com>

2016/05/16 v1.17

Abstract

Package \texttt{kvsetkeys} provides \texttt{\kvsetkeys}, a variant of package \texttt{keyval}'s \texttt{\setkeys}. It allows to specify a handler that deals with unknown options. Active commas and equal signs may be used (e.g. see babel's shorthands) and only one level of curly braces is removed from the values.

Contents

1 Documentation ........................................ 2
  1.1 Motivation ...................................... 2
  1.2 Normalizing key value lists ...................... 3
  1.3 Parsing key value lists ........................... 3
  1.4 Processing key value pairs ...................... 4
    1.4.1 Processing similar to \texttt{keyval} ............ 4
    1.4.2 Processing similar to \texttt{\setkeys*} of package \texttt{xkeyval} .......... 4
  1.5 Default family handler .......................... 5
  1.6 Put it all together ................................ 5
  1.7 Comma separated lists ......................... 6

2 Example .............................................. 6

3 Implementation ..................................... 7
  3.1 Identification .................................. 7
  3.2 Package loading .................................. 9
  3.3 Check for ε-TpX .................................. 10
  3.4 Generic help macros .............................. 10
  3.5 Normalizing ...................................... 10
  3.6 Parsing key value lists ........................... 13
  3.7 Parsing comma lists ............................... 14
  3.8 Processing key value pairs ...................... 14
  3.9 Error handling ................................... 17
  3.10 Do it all ........................................ 17

4 Test .................................................. 18
  4.1 Catcode checks for loading ...................... 18
  4.2 Macro tests ..................................... 20
    4.2.1 Preamble ..................................... 20
    4.2.2 Time ......................................... 20
    4.2.3 Test sets ..................................... 21
  4.3 Tests for key value processing handler ......... 24

∗Please report any issues at https://github.com/ho-tex/oberdiek/issues
1 Documentation

First I want to recommend the very good review article “A guide to key-value methods” by Joseph Wright [1]. It introduces the different key-value packages and compares them.

1.1 Motivation

\kvsetkeys serves as replacement for keyval’s \setkeys. It basically uses the same syntax. But the implementation is more robust and predictable:

Active syntax characters: Comma ‘,’ and the equals sign ‘=’ are used inside key value lists as syntax characters. Package keyval uses the catcode of the characters that is active during package loading, usually this is catcode 12 (other). But it can happen that the catcode setting of the syntax characters changes. Especially active characters are of interest, because some language adaptations uses them. For example, option turkish of package babel uses the equals sign as active shorthand character. Therefore package kvsetkeys deals with both catcode settings 12 (other) and 13 (active).

Brace removal: Package keyval’s \setkeys removes up to two levels of curly braces around the value in some unpredictable way:

\setkeys{fam}{key={value}} \rightarrow value
\setkeys{fam}{key={value}}} \rightarrow \{value\}
\setkeys{fam}{key={value}}} \rightarrow \{value\}
This package \texttt{kvsetkeys} follows a much stronger rule: Exactly one level of braces are removed from an item, if the item is surrounded by curly braces. An item can be a the key value pair, the key or the value.

\begin{verbatim}
\kvsetkeys{fam}{key={value}} \rightarrow value
\kvsetkeys{fam}{key={{value}}} \rightarrow \{value\}
\end{verbatim}

\textbf{Arbitrary values:} Unmatched conditionals are supported.

Before I describe \texttt{kvsetkeys} in more detail, first I want to explain, how this package deals with key value lists. For the package also provides low level interfaces that can be used by package authors.

\section*{1.2 Normalizing key value lists}

\\texttt{\kv@normalize{⟨key value list⟩}}

If the user specifies key value lists, he usually prefers nice formatted source code, e.g.:

\begin{verbatim}
\hypersetup{
  pdftitle = {...},
  pdfsubject = {...},
  pdfauthor = {...},
  pdfkeywords = {...},
  ...
}
\end{verbatim}

Thus there can be spaces around keys, around = or around the value. Also empty entries are possible by too many commas. Therefore these spaces and empty entries are silently removed by package \texttt{keyval} and this package. Whereas the contents of the value can be protected by curly braces, especially if spaces or commas are used inside, a key name must not use spaces or other syntax characters.

\begin{verbatim}
\kv@normalize{⟨key value list⟩}
\end{verbatim}

\texttt{\kv@normalize} takes a key value list and performs the cleanup:

\begin{itemize}
  \item Spaces are removed.
  \item Syntax characters (comma and equal sign) that are active are replaced by the same characters with standard catcode. (Example: \texttt{babel}'s language option \texttt{turkish} uses the equal sign as active shorthand character.)
\end{itemize}

The result is stored in \texttt{\kv@list}, e.g.:

\begin{verbatim}
\kv@list \rightarrow ,pdftitle={...},pdfsubject={...},...
\end{verbatim}

Curly braces around values (or keys) remain untouched.

\texttt{v1.3+}: One comma is added in front of the list and each pair ends with a comma. Thus an empty list consists of one comma, otherwise two commas encloses the list. Empty entries other than the first are removed.

\texttt{v1.0 – v1.2}: Empty entries are removed later. In fact it adds a comma at the begin and end to protect the last value and an easier implementation.

\section*{1.3 Parsing key value lists}

\begin{verbatim}
\kv@parse{⟨key value list⟩}{⟨processor⟩}
\end{verbatim}

It is easier to parse a normalized list, thus \texttt{\kv@parse} normalizes the list and calls \texttt{\kv@parse@normalized}.
Now the key value list is split into single key value pairs. For further processing
the key and value are given as arguments for the \langle processor \rangle:
\langle processor \rangle \{ \langle key \rangle \} \{ \langle value \rangle \}

Also key and value are stored in macro names:

- \kv@key stores the key.
- \kv@value stores the value or if the value was not specified it has the meaning \relax.

The behaviour in pseudo code:

\begin{verbatim}
foreach ((key), (value)) in ((key value list))
  \kv@key := \langle key \rangle
  \kv@value := \langle value \rangle
\end{verbatim}

Since version 2011/03/03 v1.11 \kv@break can be called inside the \langle processor \rangle
of \kv@parse or \kv@parse@normalized, then the processing is stopped and the
following entries discarded.

## 1.4 Processing key value pairs

Key value pairs can be processed in many different ways. For example, the pro-
cessor for \kvsetkeys works similar to \setkeys of package keyval. There unknown
keys raise an error.

Package xkeyval also knows a star form of \setkeys that stores unknown keys
in an internal macro for further processing with \setrmkeys and similar macros.
This feature is covered by processor \kv@processor@known.

### 1.4.1 Processing similar to keyval

\kv@processor@default \{(family)\} \{\langle key \rangle\} \{\langle value \rangle\}

There are many possibilities to process key value pairs. \kv@processor@default is
the processor used in \kvsetkeys. It reimplments and extends the behaviour of
keyval’s \setkeys. In case of unknown keys \setkeys raise an error. This processor,
however, calls a handler instead, if it is provided by the family. Both \langle family \rangle and
\langle key \rangle may contain package babel’s shorthands (since 2011/04/07 v1.13).

Since 2011/10/18 v1.15 the family handler can reject the successful handling
of a key by calling \kv@handled@false.

Since 2016/05/16 v1.17 \kv@processor@default also defines macro \kv@fam
with meaning \langle family \rangle for convenience.

### 1.4.2 Processing similar to \setkeys* of package xkeyval

\kv@processor@known \{(family)\} \{\langle cmd \rangle\} \{\langle key \rangle\} \{\langle value \rangle\}

The key value processor \kv@processor@known behaves similar to \kv@processor@default. If the \langle key \rangle exists in the \langle family \rangle its code is called, otherwise the
family handler is tried. If the family handler is not set or cannot handle the key,
the unknown key value pair is added to the macro \langle cmd \rangle. Since 2011/10/18 v1.15.

The behaviour in pseudo code:
if ⟨key⟩ exists
   call the keyval code of ⟨key⟩
else
   if ⟨handler⟩ for ⟨family⟩ exists
      handled = true
      ⟨handler⟩ {⟨key⟩} {⟨value⟩}
   if handled
      else
         add "{⟨key⟩}={⟨value⟩}" to {⟨cmd⟩}
   fi
   else
      add "{⟨key⟩}={⟨value⟩}" to {⟨cmd⟩}
      raise unknown key error
   fi
fi

Since 2016/05/16 v1.17 \kv@processor@known also defines macro \kv@fam with meaning ⟨family⟩ for convenience.

1.5 Default family handler
\kv@processor@default calls ⟨handler⟩, the default handler for the family, if the key does not exist in the family. The handler is called with two arguments, the key and the value. It can be defined with \kv@set@family@handler:

\kv@set@family@handler{⟨family⟩}{⟨handler definition⟩}

This sets the default family handler for the keyval family ⟨family⟩. Inside ⟨handler definition⟩ #1 stands for the key and #2 is the value. Also \kv@key and \kv@value can be used for the key and the value. If the value is not given, \kv@value has the meaning \relax.

\kv@unset@family@handler{⟨family⟩}

It removes the family handler for ⟨family⟩. Since 2011/10/18 v1.15.

1.6 Put it all together

\kvsetkeys{⟨family⟩}{⟨key value list⟩}

Macro \kvsetkeys processes the ⟨key value list⟩ with the standard processor \kv@processor@default:

\kv@parse{⟨key value list⟩}{\kv@processor@default{⟨family⟩}}

\kvsetknownkeys{⟨family⟩}{⟨cmd⟩}{⟨key value list⟩}

Macro \kvsetknownkeys processes the ⟨key value list⟩ with processor \kv@processor@known. All key value pairs with keys that are not known in ⟨family⟩ are stored in macro ⟨cmd⟩. A previous contents of macro ⟨cmd⟩ will be overwritten. If all keys can be handled, ⟨cmd⟩ will be empty, otherwise it contains a key value list of unhandled key value pairs. Since 2011/10/18 v1.15.

Pseudo code:

create macro ⟨cmdaux⟩ with unique name (inside the current group)
\def{cmdaux}{}
\let\cmd=\cmdaux
\\kvsetkeys@expandafter{\{}\cmd\}\\kvsetknownkeys@expandafter{\{}\cmd\}\}

Both macros behave like the counterparts without suffix \expandafter. The difference is that the key value list is given as macro that is expanded once. Since 2011/10/18 v1.15.

Thus you can replace \setkeys of package keyval by the key value parser of this package:

\renewcommand*{\setkeys}{\kvsetkeys}
or
\let\setkeys=\kvsetkeys

1.7 Comma separated lists

Since version 2007/09/29 v1.3 this package also supports the normalizing and parsing of general comma separated lists.

\comma@normalize{\comma list}

Macro \comma@normalize normalizes the comma separated list, removes spaces around commas. The result is put in macro \comma@list.

\comma@parse{\comma list}{\processor}

Macro \comma@parse first normalizes the comma separated list and then parses the list by calling \comma@parse@normalized.

\comma@parse@normalized{\comma list}{\processor}

The list is parsed. Empty entries are ignored. \processor is called for each non-empty entry with the entry as argument:

\processor\entry

Also the entry is stored in the macro \comma@entry.

\comma@break

Since version 2011/03/03 v1.11 \comma@break can be called inside the \processor of \comma@parse or \comma@parse@normalized, then the processing is stopped and the following entries discarded.

2 Example

The following example prints a short piece of HTML code using the tabbing environment for indenting purpose and a key value syntax for specifying the attributes of an HTML tag. The example illustrates the use of a default family handler.

1 (*example)
2 \documentclass{article}
3 \usepackage[T1]{fontenc}
4 \usepackage{kvsetkeys}
5 \usepackage{keyval}
3 Implementation

3.1 Identification

Reload check, especially if the package is not used with \LaTeX.
3.2 Package loading

\begin{Verbatim}
\end{verbatim}
3.3 Check for $\varepsilon$-\TeX

\texttt{\unexpanded, \ifcsname, and \unless are used if found.}

\begin{verbatim}
\begingroup\expandafter\endgroup
\ifcase0\ifetex@unexpanded
\expandafter\ifx\csname ifcsname\endcsname\relax
\else
\expandafter\ifx\csname unless\endcsname\relax
\else
1%
\fi
\fi
\catcode`$=9 % ignore
\catcode`&=14 % comment
\else % e-\TeX
\catcode`$=14 % comment
\catcode`&=9 % ignore
\fi
\end{verbatim}

3.4 Generic help macros

\texttt{\KVS@Empty}
\begin{verbatim}
\def\KVS@Empty{}
\end{verbatim}

\texttt{\KVS@FirstOfTwo}
\begin{verbatim}
\long\def\KVS@FirstOfTwo#1#2{#1}
\end{verbatim}

\texttt{\KVS@SecondOfTwo}
\begin{verbatim}
\long\def\KVS@SecondOfTwo#1#2{#2}
\end{verbatim}

\texttt{\KVS@IfEmpty}
\begin{verbatim}
\long\def\KVS@IfEmpty#1{\% & \edef\KVS@Temp{\etex@unexpanded{#1}}\%
\begingroup
\toks@{#1}\%
\edef\KVS@Temp{\the\toks@}\%
\expandafter\endgroup
\ifx\KVS@Temp\KVS@Empty \expandafter\KVS@FirstOfTwo \else \expandafter\KVS@SecondOfTwo \fi
\fi
\end{verbatim}

\texttt{\kv@normalize}
\begin{verbatim}
\long\def\kv@normalize#1{\% \begingroup
\toks@{,#1,}\%
\KVS@Comma\%
\KVS@SpaceComma\%
\KVS@CommaSpace\%
\KVS@CommaComma\%
\KVS@Equals\%
\KVS@SpaceEquals\%
\KVS@EqualsSpace\%
\xdef\KVS@Global{\the\toks@}\%
\endgroup
\let\kv@list\KVS@Global
\}
\end{verbatim}

3.5 Normalizing
\def\comma@normalize{\begingroup\toks@{,#1,}\KVS@Comma\KVS@SpaceComma\KVS@CommaSpace\xdef\KVS@Global{\the\toks@}\endgroup\let\comma@list\KVS@Global}\}

\KVS@Comma Converts active commas into comma with catcode other. Also adds a comma at the end to protect the last value for next cleanup steps.
\def\KVS@Comma{\toks@\expandafter{\expandafter}\KVS@@Comma\the\toks@~\KVS@Nil}\long\def\KVS@@Comma#1~#2\KVS@Nil{\toks@\expandafter\KVS@@Comma\the\toks@#1}{}\long\def\KVS@@Comma#1, #2\KVS@Nil{\KVS@IfEmpty{#2}{\toks@{#1}}{\KVS@@Comma#1,#2\KVS@Nil}}\KVS@SpaceComma Removes spaces before the comma, may add commas at the end.
\def\KVS@SpaceComma{\expandafter\KVS@@SpaceComma\the\toks@#1}\long\def\KVS@@SpaceComma#1, #2\KVS@Nil{\KVS@IfEmpty{#2}{\toks@{#1}}{\KVS@@SpaceComma#1,#2\KVS@Nil}}\KVS@CommaSpace Removes spaces after the comma, may add commas at the end.
\def\KVS@CommaSpace{\expandafter\KVS@@CommaSpace\the\toks@, \KVS@Nil}{\KVS@@CommaSpace#1, #2\KVS@Nil}{\KVS@@CommaSpace#1, #2\KVS@Nil}
\KVS@CommaComma Replaces multiple commas by one comma.
\begin{verbatim}
def\KVS@CommaComma{\expandafter\KVS@@CommaComma\the\toks@,\KVS@Nil}
\end{verbatim}

\KVS@CommaComma\long\def\KVS@CommaComma#1,,#2\KVS@Nil{\begin{verbatim}
\KVS@IfEmpty{#2}{\toks@{#1,}}{\KVS@@CommaComma#1,#2\KVS@Nil}
\end{verbatim}

\KVS@Equals Converts active equals signs into catcode other characters.
\begin{verbatim}
\begingroup
\lccode`\==`\=%
\lccode`\~=`\=%
\lowercase{\endgroup}
def\KVS@Equals{\toks@{\expandafter}{\expandafter}
\KVS@@Equals\the\toks@~\KVS@Nil}
\end{verbatim}

\KVS@SpaceEquals Removes spaces before the equals sign.
\begin{verbatim}
def\KVS@SpaceEquals#1{\def\KVS@SpaceEquals{\expandafter\KVS@@SpaceEquals\the\toks@#1=\KVS@Nil}}
\end{verbatim}

\KVS@SpaceEquals\long\def\KVS@SpaceEquals#1=#2\KVS@Nil{\begin{verbatim}
\KVS@IfEmpty{#2}{\toks@{#1}}{\KVS@@SpaceEquals#1=#2\KVS@Nil}
\end{verbatim}

\KVS@EqualsSpace Removes spaces after the equals sign.
\begin{verbatim}
def\KVS@EqualsSpace{\expandafter\KVS@@EqualsSpace\the\toks@= \KVS@Nil}
\end{verbatim}

\KVS@EqualsSpace\long\def\KVS@EqualsSpace#1=#2\KVS@Nil{\begin{verbatim}
\KVS@IfEmpty{#2}{\toks@{#1}}{\KVS@@EqualsSpace#1=#2\KVS@Nil}
\end{verbatim}
3.6 Parsing key value lists

`\kv@parse` Normalizes and parses the key value list. Also sets `\kv@list`.

\begin{verbatim}
\long\def\kv@parse#1{\kv@normalize{#1}\expandafter\kv@parse@normalized\expandafter{\kv@list}}
\end{verbatim}

`\kv@parse@normalized` #1: key value list
#2: processor

\begin{verbatim}
\long\def\kv@parse@normalized#1#2{\KVS@Parse#1,\KVS@Nil{#2}}
\end{verbatim}

`\KVS@Parse` #1, #2: key value list
#3: processor

\begin{verbatim}
\long\def\KVS@Parse#1,#2\KVS@Nil#3{\KVS@IfEmpty{#1}{\let\KVS@MaybeBreak\relax}\KVS@Process#1=\KVS@Nil{#3}\KVS@MaybeBreak\KVS@IfEmpty{#2}{\KVS@Parse#2\KVS@Nil{#3}}}{}
\end{verbatim}

`\KVS@Process` #1: key
#2: value, =
#3: processor

\begin{verbatim}
\long\def\KVS@Process#1=#2\KVS@Nil#3{&\edef\kv@value{\etex@unexpanded{#2}}\xdef\KVS@Global{\the\toks@}\let\kv@value\KVS@Global}
\end{verbatim}


3.7 Parsing comma lists

\commaparse Normalizes and parses the key value list. Also sets \comma@list.
\comma@parse@normalized #1: comma list
#2: processor
\KVS@CommaParse #1,#2\KVS@Nil#3: processor
\commabreak

3.8 Processing key value pairs

\kv@handled@false The handler can call \kv@handled@false or \kv@handled@true so report failure
or success. The default is success (compatibility for versions before 2011/10/18
v1.15).
\kv@handled@false
\def\kv@handled@true{\let\ifkv@handled@\iftrue}
\ifkv@handled@
\kv@handled@true
\kv@processor@default
\def\kv@processor@default#1#2{\begingroup\csname @safe@activestrue\endcsname\let\ifincsname\iftrue\edef\KVS@temp{\endgroup
\noexpand\KVS@ProcessorDefault{#1}{#2}%%}
\KVS@temp}
\KVS@ProcessorDefault
\long\def\KVS@ProcessorDefault#1#2#3{\def\kv@fam{#1}& \unless\ifcsname KV@#1@#2\endcsname
$\begingroup\expandafter\expandafter\expandafter\endgroup
$\expandafter\ifx\csname KV@#1@#2\endcsname\relax
& \unless\ifcsname KVS@#1@handler\endcsname
$\begingroup\expandafter\expandafter\expandafter\endgroup
$\expandafter\ifx\csname KVS@#1@handler\endcsname\relax
\kv@error@unknownkey(#1)(#2)%%
\else
\kv@handled@true
\csname KVS@#1@handler\endcsname(#2)(#3)%%
\relax
\ifkv@handled@
\else
\kv@error@unknownkey(#1)(#2)%%
\fi
\fi
\else
\ifx\kv@value\relax
& \unless\ifcsname KV@#1@#2@default\endcsname
$\begingroup\expandafter\expandafter\expandafter\endgroup
$\expandafter\ifx\csname KV@#1@#2@default\endcsname\relax
\kv@error@novalue(#1)(#2)%%
\else
\csname KV@#1@#2@default\endcsname\relax
\fi
\else
\csname KV@#1@#2\endcsname(#3)%%
\fi
\fi}
\kv@processor@known
\def\kv@processor@known#1#2#3{\begingroup\csname @safe@activestrue\endcsname\let\ifincsname\iftrue\edef\KVS@temp{\endgroup
\noexpand\KVS@ProcessorKnown(#1)\noexpand#2(#3)%%}
}%
\KVS@ProcessorKnown
\long\def\KVS@ProcessorKnown#1#2#3#4{\%
  \def\kv@fam{#1}\%
  \unless\ifcsname KV@#1@#3\endcsname
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname KV@#1@#3\endcsname\relax
  \unless\ifcsname KVS@#1@handler\endcsname\relax
  \KVS@AddUnhandled#2{#3}{#4}\%
  \else
  \kv@handled@true
  \csname KVS@#1@handler\endcsname{#3}{#4}\relax
  \ifkv@handled@
  \else
  \KVS@AddUnhandled#2{#3}{#4}\%
  \fi
  \else
  \ifx\kv@value\relax
  \unless\ifcsname KV@#1@#2@default\endcsname
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname KV@#1@#3@default\endcsname\relax
  \kv@error@novalue{#1}{#3}\%
  \else
  \csname KV@#1@#3@default\endcsname\relax
  \fi
  \else
  \csname KV@#1@#3\endcsname{#4}\fi
  \fi
  \fi
  \fi
  \fi
  \fi
  \fi
  \fi
  \fi
}\%
\KVS@AddUnhandled
\long\def\KVS@AddUnhandled#1#2#3{\%
  \edef#1{\%
    \ifx#1\KVS@empty
      \else
        \etex@unexpanded{#1},\%
      \fi
    \etex@unexpanded{{#2}={#3}}\%
  \}
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \ifx#1\KVS@empty
    \toks@{{#2}={#3}}\%
  \else
    \toks@\expandafter{#1,{#2}={#3}}\%
  \fi
  \xdef\KVS@Global{\the\toks@}\%
  \endgroup
  \let#1\KVS@Global\%
}\%
\kv@set@family@handler
\long\def\kv@set@family@handler#1#2{\%
  \begingroup
    \csname @safe@activestrue\endcsname
  \endgroup
  \ifx#1\KVS@empty
    \else\etex@unexpanded{#1},\%
  \else
    \etex@unexpanded{{#2}={#3}}\%
  \etex@unexpanded{(#2)=(#3)}\%
  \}
  \begingroup
    \ifx#1\KVS@empty
      \else\xdef\KVS@Global{\the\toks@}\%
      \endgroup
      \let#1\KVS@Global\%
    \endgroup
    \else
      \xdef\KVS@Global{\the\toks@}\%
      \endgroup
      \let#1\KVS@Global\%
    \endgroup
  \}
3.9 Error handling

\kv@error@novalue
\def\kv@error@novalue{% 
  \kv@error@generic{No value specified for}% 
%}

\kv@error@unknownkey
\def\kv@error@unknownkey{% 
  \kv@error@generic{Undefined}% 
%}

\kv@error@generic
\def\kv@error@generic#1#2#3{% 
  \@PackageError{kvsetkeys}{#1 key `#3'}{\MessageBreak The keyval family of the key `#3' is `#2'.\MessageBreak The setting of the key is ignored because of the error.\MessageBreak} \@ehc 
% }

3.10 Do it all

\kvsetkeys
\long\def\kvsetkeys#1#2{% 
  \kv@parse{#2}{\kv@processor@default{#1}} 
%}

\kvsetkeys@expandafter
\def\kvsetkeys@expandafter#1#2{% 
  \expandafter\kv@parse\expandafter{#2}{\kv@processor@default{#1}} 
%}

\KVS@cmd
\def\KVS@cmd{0}

\KVS@cmd@inc
\def\KVS@cmd@inc{% 
  \edef\KVS@cmd{\the\numexpr\KVS@cmd+1} 
& \begingroup 
  \count255=\KVS@cmd\relax 
  \advance\count255 by 1\relax
%}
\KVS@cmd@dec
\def\KVS@cmd@dec{
& \edef\KVS@cmd{\the\numexpr\KVS@cmd-1}\
$\begingroup$
\count255=\KVS@cmd\relax
\advance\count255 by -1\relax
\edef\x{\endgroup
\noexpand\def\noexpand\KVS@cmd{\number\count255}\
}$\x$
}$\x$
}

\KVS@empty
\def\KVS@empty{}

\kvsetknownkeys
\def\kvsetknownkeys{
\expandafter\KVS@setknownkeys\csname KVS@cmd\KVS@cmd\endcsname{}
}

\KVS@setknownkeys
\long\def\KVS@setknownkeys#1#2#3#4#5{
\let#1\KVS@empty
\KVS@cmd@inc#2\kv@parse#2{#5}{\kv@processor@known{#3}#1}
\KVS@cmd@dec
\let#4=#1%
\let#4=#1%
\KVS@setknownkeys@expandafter
\def\kvsetknownkeys@expandafter{\expandafter\KVS@setknownkeys\csname KVS@cmd\KVS@cmd\endcsname\expandafter}
\KVS@AtEnd%

4 Test
4.1 Catcode checks for loading
\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\catcode`\@=11 %
\expandafter\ifx\csname count@\endcsname\relax
\countdef\count@=255 %
\fi
\expandafter\ifx\csname @gobble\endcsname\relax
\long\def\@gobble#1{}%
\fi
4.2 Macro tests

4.2.1 Preamble

\NeedsTeXFormat{LaTeX2e}
\nofiles
\documentclass{article}
\begin{document}
\makeatletter
\chardef\KVS@TestMode=1
\makeatother
\usepackage{kvsetkeys}[2016/05/16]
\usepackage{qstest}
\IncludeTests{*}
\LogTests{log}{*}{*}
\makeatletter
\usepackage{lipsum}
\end{document}
\end{document}
\endinput
\newcommand*{\TimeDescription}{}% 
\newcommand*{\StopTime}{% 
\TestTime=\pdfelapsedtime 
\global\advance\SummaryTime\TestTime 
\PrintTime\TimeDescription\TestTime 
}% 
\let\saved@qstest\qstest 
\let\saved@endqstest\endqstest 
\def\qstest#1#2{% 
\saved@qstest{#1}{#2}% 
\StartTime{#1}% 
}% 
\def\endqstest{% 
\StopTime 
\saved@endqstest 
}% 
\AtEndDocument{% 
\PrintTime{summary}\SummaryTime 
}% 
\makeatother 

4.2.3 Test sets

\makeatletter 
\def\@makeactive#1{% 
\catcode`#1=13\relax 
}% 
\@makeactive, 
\def{\errmessage{COMMA}} 
\@makeother, 
\@makeactive= 
\def={\errmessage{EQUALS}} 
\@makeother= 

\begin{qstest}{normalize}{normalize,active-chars,space-removal}% 
\long\def\Test#1#2{% 
\@makeother, 
\@makeother= 
\scantokens{\toks@={#2}}% 
\edef\Result{\the\toks@}% 
\@makeother, 
\@makeother= 
\@Test{#1}% 
\@makeactive, 
\@Test{#1}% 
\@makeactive= 
\@Test{#1}% 
\@makeother, 
\@Test{#1}% 
\@makeother= 
}% 
\long\def\@Test#1{% 
\scantokens{\kv@normalize{#1}}% 
\expandafter\expandafter\expandafter\Expect 
\expandafter\expandafter\expandafter\kv@list\kv@list{\Result} 
\Expect*{\ifx\kv@list\Result true\else false\fi}{true}% 
}% 
\Test{}{,}% 
\Test{,}{,}% 
\Test{,,}{,}% 
\Test{,,,}{,}% 
\Test{ , }{,}%
\def\TestSet#1{\Test{#1#1}{,}\\\Test{#1#1,#1#1}{,}\\\Test{#1#1,#1#1,#1#1}{,}\\\Test{#1#1#1#1#1}{,}\\\Test{{a}#1#1=#1#1{b}}{,{a}={b},}\\}%
\TestSet{ }%\begingroup
\let\saved@normalize\kv@normalize
\def\kv@normalize#1{\saved@normalize{#1}\@onelevel@sanitize\kv@list\@onelevel@sanitize\Result}%
\Test{#,#=#,{#}={#},{#}=,{#}}{,#,#=#,{#}={#},{#}=,{#},}\\%
\endgroup
\begingroup
\def\Test#1#2{\edef\Result{#2}\@Test{#1}}%\Test{{ a = b }}{,{ a = b },}\\%\Test{{,}}{\string,{
oexpand,}\string,}\\%\Test{a={=}}{,a\string={
oexpand=},}\\%\Test{a=b}{,a=b,}\\%\Test{a={b}}{,a={b},}\\%\Test{a ={b}}{,a={b},}\\%\Test{a= {b}}{,a={b},}\\%\Test{a = {b} ,}{,a={b},}\\%\Test{a}{,a,}\\%\Test{ a}{,a,}\\%\Test{a }{,a,}\\%\Test{, a ,}{,a,}\\%\Test{, a b ,}{,a b,}\\%\Test{,a ,}{,a,}\\%\Test{ a =}{,a=,}\\%\Test{ a = }{,a=,}\\%\Test{a =}{,a=,}\\%\Test{{a} =}{,{a}=,}\\%\Test{{a}= {}}{,{a}={},}\\%\Test{, a = {}}{,a={},}\\%\Test{a,,b}{,a,b,}\\%\Test{a=\fi}{,a=\fi,}\\%\Test{a=\iffalse}{,a=\iffalse,}%
\endgroup
```latex
\Test{a=\iffalse,b=\fi}{a=\iffalse,b=\fi,}%
\end{qstest}
\begin{qstest}{parse}{parse,brace-removal}
\def\Processor#1#2{% 
  \expandafter\Expect\expandafter{\kv@key}{#1}%
  \toks@{#2}%
  \edef\x{\the	oks@}%
  \ifx\kv@value\relax
    \Expect*{\the\toks@}{%}
    \edef\Value{<>}%
  \else
    \edef\Value{[\the\toks@]}%
    \@onelevel@sanitize\Value
  \fi
  \toks@{#1}%
  \ifx\Result\@empty
    \edef\Result{[\the\toks@]=\Value}%
  \else
    \edef\Result{\Result,[\the\toks@]=\Value}%
  \fi
  \@onelevel@sanitize\Result
}%
\def\Test#1#2{% 
  \sbox0{% 
    \let\Result\@empty
    \kv@parse{#1}\Processor
    \Expect*{\Result}{#2}%
  }%
  \Expect*{\the\wd0}{0.0pt}%
}%
\Test{}{}
\Test{{}}{}
\Test{{{}}}{[{}]=<>}%
\Test{{{{}}}}{[{}]=<>}%
\Test{a}{[a]=<>}%
\Test{{a}}{[a]=<>}%
\Test{{{a}}}{{a}}{[a]=<>}%
\Test{a=}{{a}}{[a]=[{}]}%
\Test{a=}{[a]=[]}%
\Test{a=b}{{a=b}}{[a]=[b]}%
\Test{a=\iffalse}{[a]=\[\iffalse\]}%
\end{qstest}
\begin{qstest}{comma}{comma,parse}
\def\Processor#1{% 
  \expandafter\Expect\expandafter{\comma@entry}{#1}%
  \toks@{#1}%
  \ifx\Result\@empty
    \edef\Result{[\the\toks@]=\Value}%
  \else
    \edef\Result{\Result,[\the\toks@]=\Value}%
  \fi
  \@onelevel@sanitize\Result
}%
\def\Test#1{%
  \let\Result\@empty
  \kv@parse{#1}\Processor
  \Expect*{\Result}{%}
}%
\Test{}{}
\Test{{}}{}
\Test{{{}}}{[{}]=<>}%
\Test{{{{}}}}{[{}]=<>}%
\Test{a}{[a]=<>}%
\Test{{a}}{[a]=<>}%
\Test{{{a}}}{{a}}{[a]=<>}%
\Test{a=}{[a]=[]}%
\Test{a=\iffalse}{[a]=\[\iffalse\]}%
\end{qstest}
```
4.3 Tests for key value processing handler

\begin{document}
\end{document}
5 Installation

5.1 Download

Package. This package is available on CTAN:\footnote{\url{http://ctan.org/pkg/kvsetkeys}}:

\texttt{CTAN:macros/latex/contrib/oberdiek/kvsetkeys.dtx} The source file.
\texttt{CTAN:macros/latex/contrib/oberdiek/kvsetkeys.pdf} Documentation.

Bundle. All the packages of the bundle ‘oberdiek’ are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

\texttt{CTAN:install/macros/latex/contrib/oberdiek.tds.zip}

\emph{TDS} refers to the standard “A Directory Structure for \TeX{} Files” (\texttt{CTAN:tds/tds.pdf}). Directories with \texttt{texmf} in their name are usually organized this way.

5.2 Bundle installation

Unpacking. Unpack the \texttt{oberdiek.tds.zip} in the TDS tree (also known as \texttt{texmf} tree) of your choice. Example (linux):

\begin{verbatim}
unzip oberdiek.tds.zip -d ~%/texmf
\end{verbatim}

Script installation. Check the directory \texttt{TDS:scripts/oberdiek/} for scripts that need further installation steps. Package \texttt{attachfile2} comes with the Perl script \texttt{pdfatfi.pl} that should be installed in such a way that it can be called as \texttt{pdfatfi}. Example (linux):

\begin{verbatim}
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
\end{verbatim}

5.3 Package installation

Unpacking. The \texttt{.dtx} file is a self-extracting docstrip archive. The files are extracted by running the \texttt{.dtx} through plain \TeX{}:

\begin{verbatim}
tex kvsetkeys.dtx
\end{verbatim}
TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as texmf tree):

- `kvsetkeys.sty` → `tex/generic/oberdiek/kvsetkeys.sty`
- `kvsetkeys.pdf` → `doc/latex/oberdiek/kvsetkeys.pdf`
- `kvsetkeys-example.tex` → `doc/latex/oberdiek/kvsetkeys-example.tex`
- `test/kvsetkeys-test1.tex` → `doc/latex/oberdiek/test/kvsetkeys-test1.tex`
- `test/kvsetkeys-test2.tex` → `doc/latex/oberdiek/test/kvsetkeys-test2.tex`
- `test/kvsetkeys-test3.tex` → `doc/latex/oberdiek/test/kvsetkeys-test3.tex`
- `test/kvsetkeys-test4.tex` → `doc/latex/oberdiek/test/kvsetkeys-test4.tex`
- `kvsetkeys.dtx` → `source/latex/oberdiek/kvsetkeys.dtx`

If you have a `docstrip.cfg` that configures and enables docstrip’s TDS installing feature, then some files can already be in the right place, see the documentation of docstrip.

5.4 Refresh file name databases

If your TeX distribution (teTeX, mikTeX, ...) relies on file name databases, you must refresh these. For example, teTeX users run `texhash` or `mktexlar`.

5.5 Some details for the interested

Unpacking with LATEX. The `.dtx` chooses its action depending on the format:

- **plain TeX**: Run docstrip and extract the files.
- **LATEX**: Generate the documentation.

If you insist on using LATEX for docstrip (really, docstrip does not need LATEX), then inform the autodetect routine about your intention:

```
l ine \set\install=y\input{kvsetkeys.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdflatex:

```
pdflatex kvsetkeys.dtx
makeindex -s gind.ist kvsetkeys.idx
pdflatex kvsetkeys.dtx
makeindex -s gind.ist kvsetkeys.idx
pdflatex kvsetkeys.dtx
```

6 Catalogue

The following XML file can be used as source for the TeX Catalogue. The elements caption and description are imported from the original XML file from the Catalogue. The name of the XML file in the Catalogue is `kvsetkeys.xml`.

```
979 <catalogue>
980  
981  
982  
983 </name>
```
This package provides \kvsetkeys, a variant of package keyval's \setkeys. It allows the user to specify a handler that deals with unknown options. Active commas and equal signs may be used (e.g. see babel)'s shorthands) and only one level of curly braces are removed from the values.

The package is part of the oberdiek bundle.

7 References


8 History

[2006/03/06 v1.0]
- First version.

[2006/10/19 v1.1]
- Fix of \kv@set@family@handler.
- Example added.

[2007/09/09 v1.2]
- Using package infwarerr for error messages.
- Catcode section rewritten.

[2007/09/29 v1.3]
- Normalizing and parsing of comma separated lists added.
- \kv@normalize rewritten.
- Robustness increased for normalizing and parsing, e.g. for values with unmatched conditionals.
- \pX is used if available.
- Tests added for normalizing and parsing.
[2009/07/19 v1.4]
  • Bug fix for \kv@normalize: unwanted space removed (Florent Chervet).

[2009/07/30 v1.5]
  • Documentation addition: recommendation for Joseph Wright’s review article.

[2009/12/12 v1.6]
  • Short info shortened.

[2009/12/22 v1.7]
  • Internal optimization (\KVS@CommaSpace, ..., \KVS@EqualsSpace).

[2010/01/28 v1.8]
  • Compatibility to ini\TeX added.

[2010/03/01 v1.9]
  • Support of \par inside values.

[2011/01/30 v1.10]
  • Already loaded package files are not input in plain \TeX.

[2011/03/03 v1.11]
  • \kv@break and \comma@break added.

[2011/04/05 v1.12]
  • Error message with recovery action in help message (request by GL).

[2011/04/07 v1.13]
  • \kv@processor@default supports package babel’s shorthands.
  • \kv@set@family@handler with shorthand support.

[2011/06/15 v1.14]
  • Some optimizations in token register uses (GL, HO).

[2011/10/18 v1.15]
  • \kv@processor@known and \kvsetknownkeys added.
  • \kvsetkeys@expandafter and \kvsetknownkeys@expandafter added.
  • Family handler can report success or failure by \kv@handled@true or \kv@handled@false.
  • \kv@unset@family@handler added.
9 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; plain numbers refer to the code lines where the entry is used.

Symbols
\# ........................................ 589, 931
\$ ........................................ 189, 192
\% ........................................ 665
\& ........................................ 190, 193
\, ........................................ 236, 237, 740, 742, 749, 753, 756, 760, 810, 812
\= ........................................ 742, 749, 753, 756, 760, 810, 812
\> ....................................... 36, 285, 286, 288, 289, 305
\@ ........................................ 590, 663, 932
\@PackageError .......................... 529, 937
\@Test ................................... 755, 757, 759, 761, 764, 807
\@UnDeFiNeD .............................. 520
\@ehc ..................................... 535, 937
\@empty ................................... 13, 854, 863, 895, 904
\@endslash ................................ 13, 16, 30
\@firstofone................................ 598, 601
\@gobble ................................... 595, 603
\@makeactive .............................. 737, 740, 743, 756, 758, 810, 813
\@makeother .............................. 742, 745, 749, 750, 753, 754, 760, 762, 812
\@onelevel@sanitize ..................... 799, 800, 851, 859, 900
\@undefined .............................. 105
\a ......................................... 37, 38, 39, 40, 41, 42, 43, 44, 664
\{ ......................................... 587, 929
\} ......................................... 588, 930
\- ......................................... 237, 286
\advance ................................... 551, 561, 628, 636, 651, 718
\aftergroup ................................ 76
\AtEndDocument .............................. 731
\begin .................................... 34, 35, 747, 841, 891, 925
\body ...................................... 607, 611
\catcode .................................. 49, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 80, 81, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 116, 117, 119
\chardef .................................. 692
\cmd ...................................... 953, 964, 967, 969
\comma@break ................................ 6, 396
\comma@entry ................................ 387, 893
\comma@list ................................ 233, 379
\comma@parse ................................ 6, 396
\comma@parse@normalized ................... 6, 396
\count ................................... 550, 551, 553, 560, 561, 563
\count@ .................................... 592, 621
\countdef ................................ 625, 627, 628, 632, 634, 635, 636, 640, 642, 645, 646, 650, 651
\countof .................................. 592
\csname .................................... 61, 68, 97, 113, 123, 163, 166, 176, 181, 183, 408, 419, 422, 426, 437, 440, 444, 450, 461, 464, 468, 479, 482, 486, 510, 513, 517, 520, 570, 582, 591, 594, 597, 600, 655, 682, 700, 977
\defined ................................... 29
\dimexpr ................................... 708
\documentclass .............................. 2, 688
\钻 ........................................ 38, 41, 43
\empty .................................... 64, 65
\end ...................................... 45, 46, 683, 839, 889, 923, 926, 977
\endname ................................... 61, 68, 97, 113, 123, 163, 166, 176, 181, 183, 408, 417, 419, 420, 422, 426, 435, 437, 440, 444, 450, 461, 464, 468, 477, 479, 482, 486, 510, 513, 517, 520, 570, 582, 591, 594, 597, 600, 655, 682, 700, 977
\endinput .................................. 76, 161
\endlinechar ................................ 51, 114, 127, 128, 138, 139, 140, 144, 146, 189, 190, 192, 193, 587, 588, 589, 590, 625, 634, 642, 646, 663, 664, 665, 738, 929, 930, 931, 932
\equal ....................................... 120, 121, 125, 126, 127, 128, 138, 139, 140, 144, 146, 189, 190, 192, 193, 587, 588, 589, 590, 625, 634, 642, 646, 663, 742, 749, 753, 756, 760, 810, 812